Applications of Hashing in Data Encryption

Hana Shalaby
Computer Science
The American University in Cairo
UID: 900223042

Abdallah Afifi
Computer Science
The American University in Cairo
UID:900225283

Adham Gohar
Computer Science
The American University in Cairo
UID:900225576

Abstract

As a means of efficiently and effectively encapsulating data within fixed-length values, hashing algorithms have come to be recognized as a cornerstone in data encryption. This article extensively reviews the diverse hashing applications within symmetric and public key encryption. To illustrate the tangible benefits of using hashing in such contexts, we highlight specific cases where it is preferred for enhanced security, such as password storage or digital signatures. Our analysis also offers insight into how these algorithms may leave some susceptibility while providing resistance against certain attack types. Examining the merits of using hashing algorithms compared to alternative data security measures like encryption without hashing or other types of cryptographic methods is a critical area of research. This review delves into an analysis that thoroughly explores these comparisons while offering guidance on how best to apply hash functions in data encryption applications. The recommendations include selecting appropriate hash methods that align with one's needs and utilizing salt and pepper techniques alongside regular monitoring.

Introduction

The digital age has significantly changed how we communicate and conduct business. However, with increased connectivity comes increased risk. The need to secure valuable information through robust measures as a shield from cyber threats remains crucial for modern communication. As people's reliance on the internet expands, their devices have become integral to their daily lives, used to store their most sensitive information. It is important to scrutinize the measures established to protect online data and guarantee that they maintain security amidst future technological advancements.

Furthermore, with a high volume of data on the web, it is increasingly critical to develop efficient storage, access, and processing approaches, given that traditional data structures like arrays provide insufficient optimization for large datasets due to their O(n) search time complexity. The development of the hashing data structure has resolved one of the greatest challenges in computer science: storing and retrieving data effectively. Due to its unique algorithmic approach, this data structure provides constant-time storage and retrieval (O(1)) in the best-case scenario. Instead of traversing through many elements as would be done with an array, with hashing, you can easily find the index position by applying the same algorithm used during storage. As a result, it is more efficient for heavy search, deletion, and insertion applications.

Data encryption is characterized as converting plaintext data into unreadable form or ciphertext through cryptographic algorithms. Conferring resistance against attackers on the internet thus makes this practice critical. It safeguards private information while enabling authorized users' exclusive access from any location globally. This plan rests on two aspects- utilizing both an encipherment algorithm and an encoding key – that combine to encrypt or decrypt valuable communication media. Maintaining data integrity and security is essential in implementing data encryption, where one of the most fundamental techniques utilized is hashing.

Preserving data integrity is done by using cryptographic functions known as hashing algorithms. These algorithms operate by converting given inputs, also known as "messages," into fixed-size strings of bytes named "hashes" or "digests." A fundamental feature is their ability to generate different hashes with minor alterations to the entered material.

One can monitor for unauthorized adjustments in implementing these processes within data encryption methodologies. Significant examples used today include SHA and MD5. Different applications and security requirements employ algorithms with distinct performance characteristics and specific properties, making each unique.

Data encryption is a popular way of preserving shared and stored data's confidentiality, integrity, and authenticity. Hashing algorithms come into play when ensuring that these objectives are met. By utilizing these algorithms as part of the process, sensitive pieces of data are safeguarded against unauthorized access while being kept reliable. To investigate the impact of hashing algorithms on securing encrypted data, this study focuses on answering the following research question: "What are ways in which different hashing techniques enhance security?" To accomplish this goal, we will examine how these algorithms relate to various forms of data encryption. Moreover, through a comparative analysis of alternative security methods and critiquing their properties within the scope of data privacy protection matters, we will highlight key features

unique to each algorithm. Finally, we will present a case study on SHA-3 and MD5 to explore potential limitations regarding time complexities between varying kinds of algorithm implementations.

Background

Hashing Algorithms

Hashing algorithms are mathematical functions that produce fixed-size outputs known as hashes or digests from input messages or plaintexts [1]. In order to maintain data integrity and prevent malicious attacks, the property of hashing algorithms is crucial. For encryption purposes, hashing algorithms utilize one-way functions that apply complex mathematical operations to input messages. Hashing functions produce a unique output for a given input and do not allow computation of the input from the output. This produces a considerably easier hash to compute than reverting it back to its original form [1]. Using hash tables provides an effective method for managing and searching through sets of key-value pairs easily and efficiently [4]. Hashing algorithms are extensively used in various data encryption and security applications for digital signature creation, authentication, and data integrity verification [3].

Additionally, these functions exhibit an avalanche effect whereby a minor change in an input message alters the hash value entirely [7]. Therefore, a slight change in the input message is easily detected when the hashes do not match. This is an important property used in online messaging platforms such as Messenger and Whatsapp to avoid external sources tampering with personal messages.

When hashed values are stored alongside their matching input messages in such tables, quick access to these original messages can be achieved while preserving security. One of the most critical properties of hashing algorithms concerning data encryption and security applications is their ability to make pre-image resistance a reality. From a computational standpoint, it is impossible to find an input message that generates some predefined hash [5].

Collisions

Another significant aspect of hashing algorithms is their collision resistance property [6]. A collision occurs when two distinct input values are entered into the hash function and output the same hash value. This issue causes a compromise in the security of the encrypted data. From a computational perspective, finding two separate messages that generate the same hash value becomes difficult. Solutions to this problem have been developed, such as linear probing, double hashing (both open addressing), and chaining (closed addressing), where in open addressing, preserving the hash value is not essential in the design, so an alternative location is provided by traversal through the data structure until an empty slot is found. On the other hand, closed addressing ensures that the hash value is preserved for all keys with the same output from the hashing algorithm

by putting them in a linked list. These processes change the time complexity of the search and insertion processes in data encryption, an area where future research is needed to optimize the algorithm. The paper will discuss solutions to the collision problem by theoretically analyzing some hashing algorithms which are "collision-resistant."

Types of Hashing Algorithms

Data encryption and security involve various hashing algorithms, each characterized by specific advantages and shortcomings. Amongst these methods is the Message Digest 5 (MD5) algorithm relied upon for tasks such as password storage or digital signatures, producing a hash value consisting of only 128 bits [6]. However, issues related to possible collision attacks diminish its reliability compared to similar solutions available today [8].

Conversely yet equally common in this field, one can find Secure Hash Algorithm 1 (SHA-1), known for generating output resulting from computations comprising of employing up to an encoding sequence being defined at either or both stages within input or output processing ranging up-to-and-including combinations using keys provided during implementation steps – a 160-bit hash value is produced [9]. In light of recent developments, concerns have emerged over the vulnerability of encryption schemes employing the Secure Hash Algorithm 1 or SHA-1. Research findings suggest that this cryptographic hash function can be compromised by collision attacks, thereby jeopardizing data security [3].

To tackle this issue, experts at the National Institute of Standards and Technology (NIST) recommend replacing SHA-1 with newer cutting-edge hashing techniques that provide top-notch levels of protection against malicious actions [10]. In response to this call for action emerged a new generation - the versatile family known as SHA-SHA2 algorithm family, including its varieties like SHA224, SHA256, SHA384, and SHA512, resulting in hash sizes ranging anywhere between 224 bits up to 512 bits.[11]. Considered significantly secure compared to MD5 and SHA-1, these algorithms witness comprehensive utilization in data encryption and security applications. In digital signature applications, SHA-2 is extensively used and certain government agencies require it to ensure secure data transmission [13].

The adoption rate of newer technologies usually depends on how well they solve problems associated with earlier options while offering additional benefits [1]. Such has been the case with SHA-3 hashing algorithms compared to its predecessor -SHA-2 [4], [7]. The new algorithm's novel approach provided excellent resistance against parallel computing attacks, which were previously challenging to protect against [2], [5]. It produces hash values ranging from 224 up to 512 bits in length, eventually proving less susceptible to cyber threats [4], [7]. The usefulness of hashing algorithms lies in their

ability to quickly and accurately verify data integrity through a distinct hash value [8].

This attribute confirms any tampering or unauthorized modifications made to the original message, as even the slightest alteration of data renders a new hash value [8]. Additionally, these algorithms possess collision resistance, meaning that finding two distinct inputs generating the same hash value is computationally impossible [12]. Preventing malicious behavior, such as creating fraudulent messages with matching hash values, prevents security breaches [14].

When examining cryptographic hashing techniques closely, one characteristic that stands out is what is commonly referred to as the "avalanche effect" [2], [9]. This means that even minor input message changes lead to significant hash value variations generated by these algorithms [2]. The inherent benefit of this is that hackers or other malicious elements looking to infiltrate systems cannot draw parallels between related information, given how dramatically things change when hashed using popular forms like MD5, which produces different 128-bit hashes [6].

Securing digital information has gained precedence now more than ever with increasing technological advancements and expanding cybercrime activities [1], [10]. A widely known method employed in encryption processes is the use of hashing algorithms that provide unique fingerprint codes for different inputs, ensuring message authentication and checking file consistency [8], [9]. Although MD5 (Message Digest 5) and its version cousin SHA-1 (Secure Hash Algorithm) were once popular within cryptographic communities worldwide, they are no longer used as security measures due to their susceptibility towards collision attacks endangering highly sensitive private data [6], [15]. The importance of employing full encryption protocols must be stressed more since they determine how protected sensitive digital assets are from potential breaches by unauthorized parties [11].

Among the widely-used hashing functions employed for this purpose is the well-known SHA-2 algorithm which is believed to offer better security measures than earlier iterations including MD5 and SHA-1 [4]. Nevertheless, recent developments within cryptography circles demonstrate doubts concerning its resiliency, particularly regarding length extension attacks given their ability to exploit specific features unique to Hash-based Message Authentication Codes (HMACs) [3], [13]. In light of these observations, NIST has developed an improved implementation known as "SHA-3" to mitigate such weaknesses [7].

The latest hashing algorithm from NIST, SHA-3 or Secure Hash Algorithm 3, has been designed with improved security and efficiency compared to its predecessor SHA-2 [7]. Utilizing an innovative mathematical approach known as sponge functions distinguishes SHA-3 from other cryptographic hash functions [10]. This advanced method fosters faster, more

flexible hashing which best fits modern-day data encryption requirements [5], [7].

Applications of Data Encryption

Symmetric key-encryption

The significance of hashing algorithms is unparalleled regarding data protection methodologies like symmetric and public key encryption methods [1]. In secret key coding schemes or symmetric key encryption mechanisms, the same set of keys is employed during decryption and encoding activities[1]. By adopting block ciphers technology to safeguard fixed-sized datasheets, Advanced Encryption Standard (AES) delivers top-notch performance as an algorithm for symmetric digestion[5]. If encoded content is accurate, authorities can verify its consistency by implementing advanced hash functions during encrypted communication processes[2].

Secure communication protocols such as SSL/TLS rely on an effective method known as message authentication code (MAC) which guarantees the unmodified state of transmitted information. To achieve this, two separate hash values are calculated: one from the original plaintext and another from its corresponding ciphertext - by comparing these values, we can make certain any changes did not occur during transmission [2][3].

Using a pair of keys for encryption and decryption characterizes public key encryption, which is also known as asymmetric key encryption [1]. Individuals who intend to send encrypted messages to the owner of the private key share their public keys with them [1]. Public-key cryptography mostly uses the RSA algorithm that leverages challenges in determining prime factors' large numbers.

One of cryptography's most common implementations is public-key encryption utilizing hashing algorithms distinguished by their application in tasks such as digital signatures and managing keys securely [1]. Generating valid digital signatures involves taking a specific data input string whose hash value is calculated before being encrypted using a backend user's secret cryptographic material. This process yields an encrypted output so that any possible recipient of this data exchange transmission may confirm its authenticity by partitioning their trusted user's publicly viewable encryption inputs into an algorithm capable of reversing well-formed ciphertext.

Digital Certificates and Digital Signatures

In the realm of key management, hashing algorithms play a crucial role in both generating and verifying digital certificates, according to literature [1]. Digital certificates hold significant value as they authenticate secure communication entities such as email servers and websites. The certificates contain essential details about the owner along with their public keys and gets signed by a trusted third-party entity known as certificate authority (CA) [1].

In order to preserve the authentication of a digital certificate, a specific mitigation strategy known as the hash calculation is employed. Utilizing its private key, the Certificate Authority (CA) creates a hash value that receives its signature [1]. Anyone accessing the CA's public key may verify this unique digital annotation [1], assuring that no modifications have occurred and that trustworthiness can be assumed about the presented certificate.

The application of hashing algorithms in data encryption extends to digital signatures [1]. A hash function is utilized to create a message digest and the document is signed with the sender's private key in this setup [8]. By computing the message's hash and checking it against the signature's message digest via the sender's public key, recipients can verify its authenticity [8]. If the computed hashes conform, then the message is deemed genuine.

By utilizing hashing algorithms in digital signatures, numerous advantages are achieved. For instance, the message's legitimacy is guaranteed as any alterations would cause a distinct hash value, invalidating the signature [8]. Additionally, using private keys guarantees non-repudiation since only the sender can produce the signature [8]. Nonetheless, one significant disadvantage of this approach is its vulnerability to collision attacks where varying messages generate an identical hash value [14]. If a hash collision occurs, then a fraudulent message can be generated by an attacker with an identical hash value as that of the initial message. As such, gaining access to a digital signature through this method enables them to fashion an imitation thereof [14].

Password Storage

In order to enhance the security of user credentials, hashing algorithms are utilized in password storage. A common approach is salting, where a salt value is randomly added to the password before being hashed. This method enhances the difficulty for attackers to crack passwords using precomputed hash tables, as each password must first be hashed separately with its unique salt value. Another approach, referred to as stretching, involves applying the hash function repeatedly, subsequently increasing the computational cost of cracking passwords.

Issues with Data Encryption Applications of Hashing Algorithms

Data encryption employing hashing algorithms offers a multitude of benefits. However, there are several drawbacks associated with them as well. For instance, there is always a chance of hash collisions taking place wherein two different messages yield identical hash values[1]. In such an eventuality, attackers may use this information to construct fraudulent messages and cause security breaches [12].

Security threats related to weak encryption methods can be potentially

harmful due to techniques used by attackers such as preimage attacks. This kind of attack enables assailants to find messages producing specific hash values [12]. Weaker hashing techniques namely MD5 and SHA-1 can easily become targets, highlighting the necessity for utilizing stronger alternatives according to studies that have illustrated their inherent weakness towards these types of intrusions [6, 15].

Regarding issues related to speed and complexity with hashing algorithms, while these methods often prove fast and effective, some scenarios may require relying on algorithms that are particularly complex or involve significant amounts of resources in order to bolster security measures [1]. One instance of this is when looking at password stretching, a mechanism used when requiring increased computational costs regarding cracking passwords via incorporation into their respective hashing functions' delay factors [2]. This type of factor can include using numerous iterations within said hash functions or taking advantage of random data introductions prior to each calculation. The overall goal with such practices is that potential attackers face delayed processes where cracking these hashed passwords would be considerably more complex due to requiring additional calculations per every attempted guess at possible password matches. A potential drawback of password stretching may need to be addressed. Specifically, utilizing this security measure can lead to longer authentication times due to increased computational requirements, mainly if a higher number of hashing function iterations are utilized in the process [2].

Data encryption utilizing hashing algorithms yields numerous advantages, such as enhanced security, efficiency, and flexibility. However, it is vital to meticulously examine the specific application and potential weaknesses before choosing and applying a hashing algorithm for data protection.

Analysis and Critique

Hashing algorithms are commonly used to secure data during transmission or storage processes. They can produce one-of-a-kind, constant-sized outputs based on changing inputs [1]. However, their ability to enhance security depends on several variables, such as choosing an appropriate hash function type, input length consideration, also application requirements[1]. This discussion aims to evaluate the strengths and weaknesses of utilizing hashing algorithms in protecting confidential information and contrasting them with other alternatives for securing data.

Strengths and Weaknesses of Hashing Algorithms

An essential benefit of using hashing algorithms in cryptography lies in their ability to resist pre-image attacks effectively [12]. These attacks involve an attacker searching for a message that matches an existing hash value [12]. With a correctly designed hash function, such an individual

would be virtually impossible to discover said message [12]. In addition, this type of algorithm offers collision resistance protections, which are vital when dealing with fraud behavior where cybercriminals may attempt to tamper with data or create fake credentials [14].

Only some types of hash functions offer the same level of security. Several of them have potential vulnerabilities, meaning they could be subject to particular attacks. For instance, hackers have found ways to exploit the MD5 hash function through collision attacks, where two messages produce identical hashing values." Similarly, the SHA-1 hash function has been shown to have weaknesses in collision resistance and is no longer recommended for use in cryptographic applications [15]. In contrast, the SHA-2 and SHA-3 families of hash functions are currently considered secure and widely used in data encryption [4,7].

A shortcoming of hashing algorithms is their vulnerability to length extension attacks, whereby a lousy actor can augment extra information to a message lacking knowledge about the original message but still generate a matching hash value. To counter this attack, it may be helpful to employ hash functions that feature built-in message authentication codes (MACs) or apply HMAC (Hash-based Message Authentication Code) constructions.

Comparison with Other Methods for Securing Data

It is common today to use hashing algorithms to secure user data. Still, choosing an alternative method could also offer an effective solution [1]. Stream ciphers like RC4 represent one possibility that does not require hashes but still maintains the confidentiality of information through encryption techniques [10]. In contrast, these same stream ciphers lack built-in measures protecting against any changes that might occur. To counteract this issue when necessary requires combining encryption techniques with hash-based methods [1].

The realm of cryptography spans beyond its traditional role in message encryption and decryption. Alternative cryptographic primitives like MACs and digital certificates are also valuable for safeguarding data [9]. By utilizing secret keys, MACs produce fixed-size tags from messages and serve as a means of verifying data authenticity. Meanwhile, digital certificates employ hashing algorithms and public key cryptography to instill trust between parties while authenticating user or device identities [8].

Best Practices for Using Hashing Algorithms in Data Encryption

To guarantee the safety of data encryption with hash functions, it is crucial to adhere to best practices that entail picking secure hashing algorithms, implementing salt and pepper methods, and staying up-to-date with recent advancements and standardizations [1,2]. Hash functions must be chosen prudently from a pool of

thoroughly scrutinized algorithms that are currently considered unhackable [1]. Applying salt and pepper techniques that add a random string to the input message before the hashing process can increase password security by thwarting attacks such as rainbow table-based incursions [2]. Staying knowledgeable about the most recent advancements in security practices and keeping up with new developments can help better secure a system [1]. Being constantly educated on newer security standards allows for effectively addressing potential cybersecurity concerns before they become serious threats. It is also instrumental in ensuring a system is adequately updated with cutting-edge methods, effectively protecting against well-known hazards.

Regarding safeguarding data encryption, the utilization of hashing algorithms has proven to be an effective measure. However, its efficacy is contingent upon various factors, such as the selection of the hash function, input length, and application context [1]. Hashing algorithms remain popularly utilized amidst other security measures due to their simplicity and efficiency [1]. To ensure encrypted data's security and integrity, adhering to best practices for using hashing algorithms is necessary. These protocols include selecting resilient hash functions against attacks [1], implementing salt and pepper methodologies to avert rainbow table or dictionary attacks [2], and staying updated with current progressions in the field [1].

Effective utilization of hashing algorithms relies heavily on picking an adequate hash function[1]. A well-informed choice considers critical factors such as individual properties relevant to diverse use cases[1]. In certain situations where time efficiency may be prioritized over security concerns or vice versa -developers could deploy swift yet lesser secure hashes or highly secure yet slower ones[1]; thus, tailor-made solutions prove integral. When considering which hashing method best fits best towards achieving your information safety objectives[1], carefully comparing performances across pertinent attributes remains vital.

In order to enhance password security, it is recommended to implement measures such as salt and pepper techniques. Such techniques involve introducing additional random data into the input before hashing, which helps to curtail dictionary and rainbow table attacks that rely on precomputed tables of commonly used passwords. By including salt and pepper in the process, creating a new hash for every individual salt value becomes necessary for attackers, making such hack attempts significantly more challenging [2].

The discovery of new threats and vulnerabilities is constant in this field; therefore, keeping pace with advancements and updated standards is imperative [1]. The SHA-1 hash function was once employed extensively until its weakness was exposed, implying that it should not be used in modern applications [15]. Maintaining a thorough awareness of contemporary

standards and updates will ensure that the preferred hashing algorithm is secure and fits advanced requirements[1].

Although hashing algorithms have many beneficial aspects concerning data encryption, they are also associated with specific deficiencies [1]. To effectively utilize hashing algorithms for data encryption, a thorough evaluation should be undertaken alongside exploring complementary protective strategies suitable for specific contexts [1]. Staying attuned to updated guidelines and adopting best practice standards are critical factors contributing to increased encrypted information integrity [1].

Case Study

For our case study on hashing input strings, we used two common hashing algorithms SHA-3 and MD5. By measuring the elapsed time each algorithm takes to hash an input string, it becomes possible to evaluate their relative efficiency levels. Based on past research, we hypothesized that MD5 will be relatively faster than SHA-3 but less secure in its permutations.

Experiment Advantages

A crucial feature implemented in this project is the C++ polymorphism feature, where the base class - Hashing - allows adaptability to the respective algorithm by providing a virtual hash function that accepts input strings and returns hashed outputs via multiple cryptographic (SHA-3 and MD5) methods. A straightforward development of additional hashing algorithms is achievable through inheritance and virtual functions, offering ease in extending and modifying. Moreover, the implementation is advantageous in measuring individual algorithm times for hashing input strings. This aspect of our analysis is crucial when deciding which hash algorithm to use, as it makes the prioritization process easier. It allows us to study the differences in the hashing algorithms' characteristics and make the appropriate choice for a particular application.

Limitations

While the implementation only utilizes two specific hashing algorithms, namely SHA-3 and MD5, this approach neglects the fact that numerous other hashing algorithms are available in the market that could be more applicable depending on the specific use case. Despite being well-studied with widespread usage, it is essential to acknowledge that newer and improved algorithms have been developed that facilitate better performance or enhance security aspects compared to these current versions. The effectiveness of hashing notwithstanding, it is only sometimes the most appropriate choice for securing data given specific applications' requirements and constraints. This is why keeping track of developments within algorithmic diversity can lead to better precision in making decisions.

Methodology and Results

To efficiently analyze the results of the case study, an understanding of what it does is needed. Our case study aimed at taking an input string, read from an input text file ,‘input.txt’, applying the SHA-3 and MD5 hashing algorithms on them, and storing them in an output text file ‘output.txt.’ To implement the SHA-3 and MD5 hashing algorithms, we included a library called Crypto++ that provided the hashing algorithm implementations[16]. We measured the time taken by each algorithm to hash the input string using the ‘chrono’ library. After repeating the experiment over multiple trials and calculating an average for varying input strings and data types to ensure more accurate results[17], our findings suggest that the MD5 algorithm is faster than SHA-3 in the hashing process, making it the optimal choice for applications that involve many user interactions. Although it may take less time, other researchers have found that SHA-3 is more secure than MD5, suggesting that its hashing algorithms are harder to decipher, making it the viable option for highly-confidential data such as credit card information.

The implementation offered an example of how hashing algorithms can be a powerful mechanism for encrypting data. The study highlighted the strengths of using these algorithms, such as speed, but also identified challenges like inefficient processes. Appropriate algorithm selection is critical for optimizing computational performance when utilizing hash-based cryptosystems. Based on the findings, people interested in applying hashing algorithms in data encryption must investigate various possible solutions involving newer schemes or other viable security models to ensure the chosen algorithm is the most efficient for their case.

Conclusion

This article focuses on how hashing algorithms enhance data encryption by examining their attributes, performance history, and imperfections while offering recommendations on best practices to adopt while using them. Among the significant benefits cryptographic hash functions offer is their ability to provide another level of security that reinforces safeguarding confidential information from unauthorized access or cyberattacks continuously. In situations where there exist numerous input messages at varied lengths and sizes, hashing algorithms thrive due to their ability to produce unique, stable hash values for varying input messages, making it suitable for securing sensitive data such as digital signatures, authentication and passwords.

The analysis has ascertained that hashing benefits public key and symmetric encryption methods. Additionally, we have recognized specific instances where hashing algorithms heighten security levels during password storage and digital signature applications. Unlike other data security methods, hashing algorithms offer distinct advantages over encryption without hashing, namely stream ciphers. These algorithms also work perfectly with cryptographic tools like MACs and digital certificates.

Nonetheless, the research discussed drawbacks correlated with applying hashing algorithms in data encryption due to susceptibility to length extension attacks and brute force attacks.

The research further outlines best practices for using hashing algorithms, emphasizing the significance of intelligent hash function selection while utilizing techniques such as salt and pepper to augment security measures. The need to explore quantum-resistant hashing algorithms is prompting researchers' attention, given current advancements in quantum computing technology.

Protecting sensitive data requires a multifaceted approach that includes implementing robust security measures such as hashing algorithms. These algorithms are crucial components of modern encryption techniques due primarily to their efficacy—and yet they are not without limitations or vulnerabilities that must be addressed through continued research efforts if we wish to ensure optimal data security across multiple disciplines.

References

[1] Menezes, A., van Oorschot, P., & Vanstone, S. (1996). Handbook of applied cryptography. CRC press.

[2] Schneier, B. (1996). Applied cryptography: protocols, algorithms, and source code in C (2nd ed.). John Wiley & Sons.

[3] Ferguson, N., & Schneier, B. (2003). Practical cryptography. John Wiley & Sons.

[4] National Institute of Standards and Technology. (2015). Federal Information Processing Standards Publication 180-4: Secure Hash Standard (SHS).

[5] National Institute of Standards and Technology. (2001). Federal Information Processing Standards Publication 197: Advanced Encryption Standard (AES).

[6] Rivest, R. L. (1992). The MD5 message-digest algorithm. RFC1321.

[7] National Institute of Standards and Technology. (2015). Federal Information Processing Standards Publication 202: SHA-3 standard: Permutation-based hash and extendable-output functions.

[8] Bellare, M., & Rogaway, P. (2006). Introduction to modern cryptography. CRC press.

[9] Katz, J., & Lindell, Y. (2014). Introduction to modern cryptography (2nd ed.). Chapman and Hall/CRC.

[10] Daemen, J., & Rijmen, V. (2002). The design of Rijndael: AES-the advanced encryption standard. Springer.

[11] Boneh, D., & Shoup, V. (2017). A graduate course in applied cryptography. Retrieved from https://crypto.stanford.edu/~dabo/cryptobook/.

[12] Rogaway, P. (2003). Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. Lecture Notes in Computer Science, 2729, 371-388.

[13] Kelsey, J., Schneier, B., & Wagner, D. (2005). Key-lengths of symmetric ciphers for different applications. Proceedings of the 4th ACM Conference on Computer and Communications Security, 3-21.

[14] Sasaki, Y. (2017). Collision resistance and its applications. Cryptography and Network Security, 98-113.

[15] Kaliski, B. S. (1994). The MD4 message-digest algorithm. RFC 1320.

[16] W. Dai, "Crypto++ Library 8.6.0," 2021. [Online]. Available: https://www.cryptopp.com/. [Accessed: 29-Apr-2023].

[17] A. Fathi, H. Shalaby, & A. Gohar "Research-Paper," 2023. [Online]. Available: https://github.com/AbdallahFathi22/Research-Paper [Accessed: 29-Apr-2023].